



# DW\_sqrt

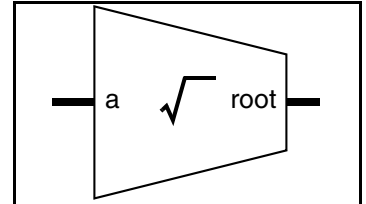
## Combinational Square Root

Version, STAR, and myDesignWare Subscriptions: [IP Directory](#)

### Features and Benefits

- Parameterized word length
- Unsigned and signed (two's complement) square root computation
- Inferable using a function call

### Revision History



### Applications

- RMS measurements
- Real-time digital signal processing
- Adaptive filtering
- Computation of gradients for edge detection

### Description

DW\_sqrt computes the integer square root of *a*. The parameter *tc\_mode* determines whether the input *a* is interpreted as unsigned (*tc\_mode* = 0) or two's complement (*tc\_mode* = 1) number.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	<i>width</i> bits	Input	Radicand
root	int([ <i>width</i> +1]/2) bits	Output	Square root

Table 1-2 Parameter Description

Parameter	Values	Description
width	≥ 2	Word length of <i>a</i>
tc_mode	0 or 1 Default: 0	Two's complement control <ul style="list-style-type: none"><li>■ 0 = Unsigned</li><li>■ 1 = Signed</li></ul>

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Restoring ripple-carry synthesis model	DesignWare
cla	Restoring carry-lookahead synthesis model	DesignWare

a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table.

**Table 1-4 Simulation Models**

Model	Function
DW02.DW_SQRT_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_sqrt_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_sqrt.v	Verilog simulation model source code

**Table 1-5 Functional Description**

tc_mode	a	root
0	a (unsigned)	$\text{int}(\sqrt{a})$ (unsigned)
1	a (two's complement)	$\text{int}(\sqrt{\text{absval}(a)})$ (unsigned)

## Application Examples

### Fixed-Point Arithmetic

For a radicand that is in *integer.fraction* format, its fraction width must be a multiple of two. The fraction width of the square root is then half the fraction width of the radicand as shown in the following example:

$$\text{sqrt}(100111.0001) = 110.01$$

### Rounding

DW\_sqrt truncates fractions to produce an integer square root. To round to the nearest integer value, concatenate two fraction bits of zero to the least significant end of the radicand. The resulting square root has a one-bit fraction. Add one-half to round the square root to the nearest integer.

### VHDL example

```
signal root_temp : unsigned(root_width downto 0);
root_temp <= (a & "00") ** one_half + 1;
root <= root_temp(root_width downto 1);
```

## Verilog example

```

wire frac;
assign {root, frac} = DWF_sqrt_uns ({a, 2'b00}) + 1;

```

## Unsigned-Signed Square Root

DW\_sqrt can only perform either unsigned or signed square root computation (specified by parameter `tc_mode`). If a square root part is required that can do both (selectable by a `tc` pin), extend the input by one bit (such as zero-extend for unsigned numbers and sign-extend for two's complement numbers depending on the `tc` pin) and compute the signed square root.

## VHDL example

```

signal root_ext : signed(root_width downto 0);
root_ext <= ((tc and a(width-1)) & a) ** one_half;
root <= root_ext(root_width-1 downto 0);

```

## Verilog example

```

wire ext_bit;
assign {ext_bit, root} = DWF_sqrt_tc ({tc & a[width-1], a});

```

## Alternative Implementation of Square Root with DW\_lp\_multifunc

The square root operation can also be implemented with the DW\_lp\_multifunc component, which evaluates the value of square root with 1 ulp error bound. There will be 1 ulp difference between the value from DW\_lp\_multifunc and the value from DW\_sqrt. Performance and area of the synthesis results are different between the DW\_sqrt and square root implementation of the DW\_lp\_multifunc, depending on synthesis constraints, library cells and synthesis environments. By comparing performance and area between the square root implementation of DW\_lp\_multifunc and DW\_sqrt component, the DW\_lp\_multifunc provides more choices for better synthesis results. Below is an example of the Verilog description for the square root of the DW\_lp\_multifunc. For more detailed information, see the [DW\\_lp\\_multifunc](#) datasheet.

```

DW_lp_multifunc #(op_width, 2) U1 (
    .a(a),
    .func(16'h0002),
    .z(z),
    .status(status)
);

```

## Related Topics

- [Math – Arithmetic Overview](#)
- [DesignWare Building Block IP User Guide](#)

## HDL Usage Through Operator Inferencing - VHDL

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use DWARE.DW_Foundation_arith.all;

entity DW_sqrt_oper is

    generic ( radicand_width : positive := 8);
    port ( radicand      : in  std_logic_vector(radicand_width-1 downto 0);
          square_root_uns : out std_logic_vector((radicand_width+1)/2-1 downto 0);
          square_root_tc  : out std_logic_vector((radicand_width+1)/2-1 downto 0));
end DW_sqrt_oper;

architecture oper of DW_sqrt_oper is
begin
    -- operators for unsigned/signed square root
    square_root_uns <= unsigned(radicand) ** one_half;
    square_root_tc  <= signed(radicand)  ** one_half;
end oper;
```

## HDL Usage Through Operator Inferencing - Verilog

Verilog has no exponentiation operator, so operator inferencing is not possible.

## HDL Usage Through Function Inferencing - VHDL

```

library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use DWARE.DW_Foundation_arith.all;

entity DW_sqrt_func is
    generic ( radicand_width : positive := 8);
    port ( radicand      : in  std_logic_vector(radicand_width-1 downto 0);
          square_root_uns : out std_logic_vector((radicand_width+1)/2-1 downto 0);
          square_root_tc  : out std_logic_vector((radicand_width+1)/2-1 downto 0));
end DW_sqrt_func;

architecture func of DW_sqrt_func is
begin
    -- function calls for unsigned/signed square root
    square_root_uns <= std_logic_vector(DWF_sqrt (unsigned(radicand)));
    square_root_tc  <= std_logic_vector(DWF_sqrt (signed(radicand)));
end func;

```

## HDL Usage Through Function Inferencing - Verilog

```

module DW_sqrt_func (radicand, square_root_uns, square_root_tc);

    parameter radicand_width = 8;

    input  [radicand_width-1 : 0]      radicand;
    output [(radicand_width+1)/2-1 : 0] square_root_uns;
    output [(radicand_width+1)/2-1 : 0] square_root_tc;

    // pass the "func_width" parameter to the inference functions
    parameter width = radicand_width;

    // Please add search_path = search_path + {synopsys_root + "/dw/sim_ver"}
    // to your .synopsys_dc.setup file (for synthesis) and add
    // +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator command line
    // (for simulation).
    `include "DW_sqrt_function.inc"

    // function calls for unsigned/signed square root
    assign square_root_uns = DWF_sqrt_uns (radicand);
    assign square_root_tc  = DWF_sqrt_tc  (radicand);

endmodule

```

## HDL Usage Through Component Instantiation - VHDL

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_sqrt_inst is
    generic (radicand_width : positive := 8;
            tc_mode         : natural   := 0);
    port (radicand      : in  std_logic_vector(radicand_width-1 downto 0);
          square_root   : out std_logic_vector((radicand_width+1)/2-1 downto 0));
end DW_sqrt_inst;

architecture inst of DW_sqrt_inst is
begin
    -- instance of DW_sqrt
    U1 : DW_sqrt
        generic map (width => radicand_width, tc_mode => tc_mode)
        port map (a => radicand, root => square_root);
end inst;

-- pragma translate_off
configuration DW_sqrt_inst_cfg_inst of DW_sqrt_inst is
    for inst
    end for;
end DW_sqrt_inst_cfg_inst;
-- pragma translate_on
```

## HDL Usage Through Component Instantiation - Verilog

```
module DW_sqrt_inst (radicand, square_root);
    parameter radicand_width = 8;
    parameter tc_mode        = 0;

    input  [radicand_width-1 : 0]      radicand;
    output [(radicand_width+1)/2-1 : 0] square_root;
    // Please add +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator
    // command line (for simulation).

    // instance of DW_sqrt
    DW_sqrt #(radicand_width, tc_mode)
        U1 (.a(radicand), .root(square_root));
endmodule
```

## Revision History

For notes about this release, see the [DesignWare Building Block IP Release Notes](#).

For lists of both known and fixed issues for this component, refer to the [STAR report](#).

For a version of this datasheet with visible change bars, click [here](#).

Date	Release	Updates
January 2020	DWBB_201912.1	<ul style="list-style-type: none"><li>Corrected port names for DW_lp_multifunc in “<a href="#">Alternative Implementation of Square Root with DW_lp_multifunc</a>” on page 3</li></ul>
July 2019	DWBB_201903.3	<ul style="list-style-type: none"><li>Removed reference to minPower library in “<a href="#">Alternative Implementation of Square Root with DW_lp_multifunc</a>” on page 3</li><li>Added this Revision History table and the document links on this page</li></ul>

## Copyright Notice and Proprietary Information

© 2022 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

### Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

### Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

### Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

### Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.  
[www.synopsys.com](http://www.synopsys.com)